NASA RESEARCH GRANT

AUTOMATED KNOWLEDGE GENERATION

FIRST YEAR FINAL REPORT

NAG10-0042

(NASA-CR-181317) AUTOMATED KNOWLEDGE GENERATION First Report, 1st Year (University of Central Florida) 17 p CSCL 09B G3/63

N89-11440

Unclas 0162009

UNIVERSITY OF CENTRAL FLORIDA
COMPUTER ENGINEERING DEPARTMENT
DR. HARLEY R. MYLER
DR. AVELINO J. GONZALEZ
SEPTEMBER 28, 1988

Davis earlier this week by Dr. Avelino Gonzalez.

I am also submitting under separate cover one bound and one unbound copy to NASA Scientific and Technical Information Facility as required by the contract.

If Mr. Davis or you have any questions or need additional information, please call Dr. Gonzalez at (407) 275-2236 or me.

Sincerely,

Betsy L. Gray
Contract Coordinator

BLG/hc

cc: Mr. Tom Davis/PT-TPO
NASA STIF

NASA FINAL REPORT

1.0 PROGRAM OBJECTIVES

The general objectives of the NASA/UCF Automated Knowledge Generation Project have been the development of an intelligent software system that could access CAD design data bases, interpret them, and generate a diagnostic knowledge base in the form of a system model. The initial area of concentration is in diagnosis of process control system using the Knowledge-based Autonomous Test Engineer (KATE) diagnostic system. A secondary objective has been the study of the general problems of automated knowledge generation.

There exist considerable obstacles in accomplishing these objectives. Dr. S. J. Thomas discusses some of these, as he developed a system to carry out some of these tasks during his summer fellowship at KSC in 1987.

The biggest problem found by the UCF team, as well as Dr. Thomas, has been that a typical CAD data base does not contain all the information necessary to generate a complete knowledge base as required by KATE. In his final report to NASA titled "Automated Construction of a Knowledge Base from Computer Aided Design Data," Thomas put forth estimates of the potential capability of an automated process to generate a knowledge base in its complete form. These estimates are shown in Table 1.

The investigation undertaken by UCF, nevertheless, has as its ultimate goal the development of a system which will be able to generate a complete knowledge base from a CAD data base, even though the latter does not contain all the information necessary to accomplish the former.

At the present time, in order to develop a knowledge base for use by KATE, a process engineer is required to review drawings of the system. Such a person would recognize the symbols as certain components, and based on his knowledge and experience, would be able to determine the function of such a component, even though is not explictly spelled out in the drawing.

Therefore, in order to create an automated knowledge generator, it is important that it possess the knowledge to enable it to provide the information not found on the CAD

data base. It is expected that such a feature, would allow the AKG to automatically generate nearly all the slots which KATE knowledge frame requires, with minimum human interface.

A secondary goal of the project was to make AKG generic so that different CAD systems could be accessed with minimal changes. It was also desirable to make this accessing capability automatic so as, once again, try to minimize human interaction.

The specific objectives of the first year of the project on which this document is based, was to develop a prototype based on object-oriented language (Flavors) which would accomplish the equivalent functionality achieved by Thomas in his work. The advantage of this prototype is that it would be done in a way so that the structure of the program would allow for ease of expansion in functionality without system re-design. This expansion would be carried out in the second and third years.

This document describes the work performed during the first year, the goals accomplished and the obstacles faced. It will also address the plans for subsequent years.

2.0 AUTOMATED KNOWLEDGE GENERATION (AKG) SYSTEM

The AKG system is designed to be a versatile frame generator system. It can take input from a Intergraph CAD system database and produce a framebased knowledge base. This section describes in detail the AKG system design and the philosophy behind some of the techniques and assumptions used in its development and implementation. Also discussed here are some problems faced and recommendations for future expert systems developments.

2.1 AKG DESIGN PHILOSOPHY

The Object Oriented Programming (OOP) approach has been taken in the development of the AKG system. It is being developed on a Symbolics 3650 LISP machine, where each component in the CAD database is represented as an object. The Flavors Object Oriented Programming facilities of the Symbolics machines are used to produce these objects. In AKG, each object is represented as a "components flavor", which consist of several instance variables (slots). These slots could be divided into two groups: 1. Slots which are used to generate the final result (frame's slot). 2. Slots which are used for the internal operations of AKG. The follwing slots are in group 1:

Nomenclature
AKO

AIO
APO
AEO
PARTS
TOLERANCE
SOURCE
CVALUE
SOURCE-PATH
IN-PATH-OF
SINKS
KINDS
INSTANCES
UNITS
IMAGES
DELAY
RANGE

The description of the above slots could be found in KATE's related paper. Group 2 slots are defined below:

NAME: This slot holds the name of a component, this information is used for the graphical presentation of a component.

DESCRIPTION: This is the description of a component in the CAD database. This could replace the content of the NOMENCLATURE slot in the frame.

FROMCONNPTS: This slot identifies all the upstream components for this component, it also holds the information concerning the connecting points between this component and its upstream components.

TOCONNPTS: This slot identifies all the downstream components for this component, it also holds the information concerning the connecting points between this component and its downstream components.

XPOS: This slot holds the X-axis location of a component in the graphical presentation of the investigated system.

YPOS: This slot holds the Y-axis location of a component in the graphical presentaion of the investigated system.

WINDOW: This slots holds the information regarding the window in which the graphical presentation is displayed.

PRESENTATION: This slot holds the information regarding the presentation type of a component. Presentation type is a Symbolics facility used by AKG to present a component as a graphical presentation.

BOX: This slot also is part of the Symbolics facilities which is used to present a component as a graphical presentation.

For more detailes on the **PRESENTATION**, **BOX**, and **WINDOW** refer to Symbolics manuals 7A, and 7B.

Besides the OOP approach, AKG system utilizes modularity techniques. As the system grows in functionality and sophistication, its modular construction allows the system developer to change part of the system without affecting other parts. In addition, the approach facilitates the incremental development of the system. AKG is currently divided into seven modules:

1. ACCESS
2. SPAWN
3. CONSTRAINT GENERATOR
4. BUILDER
5. RESOLVER
6. COMPONENT & PROCESS DATABASE
7. USER INTERFACE

All the modules with the exception of the RESOLVER are either completed or in the process of development. The following sections describe each module in the AKG system and their current status.

ACCESS

The purpose of this module is to enable AKG (or AKG users) to access remote computers in which the CAD database resides. This module makes available all the necessary facilities for the user to access any remote computer by two simple commands (actually two mouse clicks). This is accomplished through special access files written for each remote computer. These files produce the login, password, and all other necessary information (control signals) which are required to access a remote computer. Currently AKG is able to connect to the Computer Engineering Departments's GOULD or VAX computers. **ACCESS** is in the initial stages of the development, Ultimately, in addition to being able to connect to any computer at any site, **ACCESS** will be able to automatically transfer files to and from these remote computers.

SPAWN

The purpose of this module is to identify all the components in a system, and generate an object (instantiate a component flavor) for each of these components in the AKG system. This is done by the SPAWN module manipulating a CAD file called compu.dat. Here it is assumed that it is an easy task to write a inquiry program on any CAD system to generate a **compu.dat** file. **SPAWN** has been completed.

CONSTRAINT GENERATOR (CG)

The objective of this module is to find the interconnectivity of each object in the system. In order to find this connectivity, the **CG** module manipulates the contents of a formatted CAD file called **tofrom.dat**. In this module each object in the **tofrom.dat** file will receive a connectivity message. The content of this message is the identity of the upstream and downstream objects for the object receiving the message. In addition to the above information, the connectivity message includes information about the connecting points on each object. Here again it is assumed that it is possible to generate a file like **tofrom.dat** in any CAD system. **CONSTRAINT GENERATOR** is in the final stage of its development.

BUILDER

The Builder routine manipulates the flavor-components from the Make-frame module (discussed in latter sections) to setup frames that would be more acceptable to the diagnostic system. Presently, Builder produces frames that are specific to KATE standards as all the frames begin with a function KATE uses called "deframe" and all follow certain syntax constraints (parentheses placements) that are apparent in KATE's frames. The completed frames are written to an output file called "kate-frames-file.lisp".

While producing these frames, Builder also makes changes in their in-path-of and source-path slots to allow for a more sound representation conducive to diagnostic needs. In doing so, two global variables are set up -- *measurement-list* and *command-list* -- which are lists of objects considered to be measurements or commands respectively. The attainment of these lists rely solely on checking the AIO slots of the components for an indication of either command or measurement. This method is only temporary since these slots are added to the component database directly from the Purge demo data. The specific method that will eventually be used is not yet designed but various methods are being discussed and are current topics in the role of the Resolver module.

An assumption was made that the CAD data of the Purge demo was typical of CAD generated files which indicated that command and measurement devices are connected as starting points in control paths, i.e. their source-path slots are nil

and their in-path-of slots are connected to components. Using this assumption, it was noted that such a setup is unacceptable for interface with a diagnostic system. Therefore, in order to follow KATE's convention that commands are in the source-path of components ("upstream") and measurements are in the in-path-of slots of components ("downstream"), the following procedure is implemented:

1. Switch the source-path and in-path-of slots of the measurement frames.

2. Using *measurement-list* to access the measurement frames and check their source-path slots, set up an association list containing the measurements and the components they affect.

3. Traversing the list, add the associated measurement to the in-path-of slots of the components it affects and delete the measurement out of the components' source-path slots.

Note that the commands are already set up properly and do not need to be changed.

RESOLVER

The objective of this module is to accept any inconsistent information for an object from any module in the AKG, and try to resolve these inconsistencies. This action is performed using relaxation labeling, which will be discussed in more detail in following sections. It is the belief of the research team that **RESOLVER** will be the most complicated module in the AKG system. This module is still in the conceptual design stage, and no code has been written for it.

USER INTERFACE

The objective of this module is to allow non-AI oriented personnel to operat AKG. It will generate a graphical presentation of the investigated system for the user. In addition to graphical presentation of the system, AKG's **USER INTERFACE**, allows the user to perform any operation (any AKG's operation) with one or two mouse clicks. Since each object in the graphic presentation is mouse sensitive, the user can click left on any object in the drawing to get the characteristics (properties) of that object. The following commands are mouse sensitive in the **USER INTERFACE**:

1. **Access**: A right or left mouse click on this command will actuate the Access module. As a result of the above operation another mouse sensitive menu is offered to the user. The options in this menu are 1. Access to VAX, 2. Access to GOULD, or 3. Exit Access. A right or left mouse

click on choice 1 or 2, will result in a connection to the VAX, or GOULD computer respectively in the UCF Engineering building. As it was discussed previously the user is not required to know the login or password for either computers. It is obvious that the third option is for exiting the access module.

2. **Builder**: A right or left mouse click on builder will instantiate the Builder module. Builder will collect all the information for each object in the system and generate Knowledge base frames (these frames will be in the KATE format). Each frame will be saved in a file called Kate-frames-file, which is located under the sys:akg;system-output directory on the symbolics machine. Currently, Builder manipulates the in-path-of and source-path slots of each object in order to generate frames which are closer to the KATE format.

3. **Clean Display**: A right or left click on this option will clean up all the AKG's windows.

4. **Constraint Generator**: A right or left click on this option instantiates the constraint generator module. As it was discussed previously, as a result of this option, all the objects in the system will find their connectivities with respect to the other objects in the system.

5. **Draw KB**: A right or left click on this option will draw the connectivities between each object in the system. This module will examine each flavor in the system and based on the present status of the flavor, it will generate a graphical representation of it. Therefore in order to have a correct presentation of the system, it is critical to perform this function after the Constraint Generator, Make Frame, and Builder operations. As it was discussed previously, each object in this drawing is mouse sensitive, and the left mouse click on an object will display its characteristics.

6. **Make Frame**: As a result of left or right click on this command, each object will attempt to define the rest of its characteristics. Some of this information is located in the CAD files (i.e. Unit, Range, etc.), and some of them are located in the Component, and Process Database. Once all this information is collected Make Frame operation is complete. Ultimately, Make Frame will have the ability to identify all the objects which have missing critical information (critical information will be defined later), and mark these objects as incomplete for further processing.

7. **Readme**: By clicking on this command, another mouse sensitive menu will pop up. This menu gives a range of readme files for each of the commands in the user interface.

TABLE 1. COMPARISON OF RESULTS.

	THOMAS' RESULTS			AKG'S RESULTS		
Slots	Filled Auto.	Percent	Est. of Pot. Cap.	Filled Auto.	Percent	Est. of Pot. Cap.
aio	52/52	100%	100%	26/26*	100%	100%
aeo	NA	NA	NA	14/14*	100%	> 75%
nomencl.	0/52	0%	100%	13/13*	100%	100%
source-path	8/52	15%	50%	25/25#	100%	> 90%
in-path-of	52/52	100%	> 90%	35/35	100%	100%
source	2/2	100%	> 80%	NA	NA	100%
tolerance	NA	NA	NA	2/2*	100%	75%
delay	0/3	0%	0%	3/3*	100%	75%
status	0/52	0%	50%	6/6*	100%	75%
units	23/23	100%	100%	23/23	100%	100%
range	16/16	100%	100%	15/15	100%	100%
sinks	2/2	100%	> 80%	NA	NA	100%

NOTES:

*: Filled with the help of the component database

#: Need special operators to get this result

of slots generated, the progress made in its connectivity and other areas has been substantial.

Some of KATE's knowledge representation conventions that AKG uses are generally confined to the type of slots needed in a single "frame" or device representation and to the information that each slot requires for the proper operation of a diagnostic system. Other conventions that are specific to KATE are reproduced by the Builder module and does not affect the outcome of other modules. Thus changing standards or conventions specific to KATE can easily be reflected in this module.

One of the significant features of a diagnostic tool such as KATE is that it is generic in nature. Providing a knowledge base for a generic diagnostic tool requires the knowledge base to be complete within itself (defining all objects and instances of those objects) and not be function dependent -- meaning it should not lend itself to establishing its own instantiations in the Lisp environment with the aid of functions. The version of KATE presently in use with the AKG project does not support function independency. KATE uses functions for various categories of device frames such as the function "analog-read" which is contained directly in the analog device frames section of the knowledge base. The method that was used to provide a transferable knowledge base necessary for testing is first for AKG to generate KATE-like frames and then to later manually include KATE-specific details. This approach preserves the generic goal and will be used until newer versions of KATE can be evaluated or until some sort of industry standard is created.

To date, the knowledge base generated by AKG contains frames that cover the full Purge demo circuit, excluding pseudo-pressures, and are complete in their connectivity. Pseudo-pressures are measurement frames that the Purge demo uses to buffer the flow between the compressors with valves circuit and the digital logic with valve circuit. They do not behave like measurement devices, however, and seem to serve no purpose other than securing points of reference for the user. A proper CAD connection between these two sub-circuits could have achieved the same results using actual measurement devices. In the abscence of useful measurement areas in the CAD data, a structured pseudo-measurement frame properly implemented could be helpful. If it were possible for AKG to identify points of interest within a circuit to enhance readability and accessibility, then a general frame could easily be set up to provide a possible location for measurement or other reference while still keeping the connectivity of the circuit intact.

The connectivity of a circuit in AKG's terms depends on the integrity of the source-path and in-path-of slots. These

slots reflect good connectivity coming from AKG's Constraint-generator, but do not reflect the correct control flow at this point. By determining the control components, i.e. measurements and commands, and completing the function set-measurements contained in Builder, near perfect slots can be generated. The limitations of these slots being syntax in the case of KATE and conventionalism in the case of other diagnosers. The convention being that commands are placed "upstream" (ref. Cornell) and measurements are placed "downstream" of affecting components. Though, this convention is a logical one.

The knowledge base that KATE currently uses can be divided basically into two parts. The first part being the actual system components along with their control mechanisms or commands and measurements (from now on the term objects will be used to refer collectively to commands, measurements and compcnents). This can be called the low level knowledge. The other part is a sort of behind-the-scenes part that can be called the high level knowledge. It consists of more general objects or descriptions of objects that are usually "ancestors" of the low level objects. It is required in the function of KATE because it provides needed information about the character of an object. Here the generic nature of KATE wanes again because along with functions in the knowledge base, there are calls to specific components in the high level knowledge base from functions in the body of KATE which would mean that if a low level knowledge base substitution were done, the high level knowledge base would need to be edited for unfamiliar circuits. And if a total knowledge base substitution were done, the high level knowledge would need to be constrained to KATE's specific details.

This high level knowledge, however, may not be absolutely necessary in future generic diagnosers because much of the information a diagnoser needs is provided within individual frames. It is essential, however, for a high level knowledge base, which would include a very great deal of information on all types of circuits and devices, to be developed in AKG to allow for the full operation of pending modules and the complete unaided generation of all frame slots including tolerances and delays as well as the high level knowledge for a particular circuit if needed.

The basics of the low level knowledge base is in part presently accomplished as far as circuit completeness is concerned. With the assumption that the command and measurement devices are known, AKG is able to produce frames with good connectivity and near perfect in-path-of and source-path slots. The development of the high level knowledge base and the complete generic frame are topics constantly discussed and a lot of good ideas have surfaced

and are soon ready to be focused into a single solid objective.

4.0 Fourth Quarter Accomplishments

The project team has succeeded in completing the AKG prototype system as described in earlier reports. To summarize, the Automated Knowledge Generation system is constructed from individual modules written in Common Lisp and running on Symbolics Lisp engines. It consists of six major modules, which perform the following functions:

Access -- queries the CAD data source producing the component and to-from lists.

Spawn -- creates an *AKG Object* for each component.

Constraint Generator -- orchestrates the labeling process of the components by propagating confidences.

Builder -- constructs a diagnostic frame for the target reasoning system from each resolved component, in this version of AKG it is KATE.

Resolver -- attempts resolution of components whose confidence fails to reach a global threshold. Performs primary access to the *Component and Process* databases.

Graphic User Interface -- allows human supervision of the AKG process.

We have been successful in performing an automated conversion of the Purge Demo system as described by a CAD database using the AKG prototype. Development of the process has revealed a number of unanticipated factors that directly impact the AKG process. These factors are specific to KATE and are all related to an overall lack of genericity. These factors are outlined here:

a) KATE uses non-generic functions within its knowledge base (KB) that require special handling. This means that AKG will need to be

handling. This means that AKG will need to be less generic and more "KATE-like".

Our solution is to remove these specifics from KATE and make KATE examine objects individually to determine interface requirements. This may require KATE to be run in a parallel environment to operate in real-time.

b) In addition to functions appearing in the KB, there are calls to specific components within the high level knowledge base from functions *within the body* of KATE's inferencing mechanism. When AKG creates a KB, it has no ability to modify KATE to account for these changes. This implies that KATE must be made more generic.

c) The status slots seem to be reliant on KATE-specific functions for boolean expressions. These must be removed and replaced with system generic functions that will pertain to a diagnostic requirement standard.

We will be focusing on solutions to these problems and will be working towards a generic diagnostic standard.

5.0 Second Year Goals

We must now focus our research on refining the AKG to operate on a "real" system, the Orbiter Maintenance and refurbishment Facility, or OMRF. The team has succeeded in producing CAD drawings of the OMRF and is working on getting a database constructed for it. ANSI standards have been procured that describe how drawings must be done. Descriptions of components following the ANSI standards will be added to the component database. Because of the complexity of system energy flow in the OMRF, it will provide a fertile foundation for the development of the process database.

The team has anticipated the need for parallel processing of both AKG and that of frame-based diagnostic systems. AKG requires parallelism due to the enormous expected size of the component and process databases that must be searched and the propagation of relaxation confidence measures across the component structure network. If KATE is redesigned to be fully generic, then KATE will have to examine each component transfer function separately in order to evaluate a malfunction. With existing implementations, this will not be

possible to do in real-time. Our solution to both of these problems is to acquire a large Serial Instruction stream, Multiple Data stream, or SIMD, machine. At this writing, a SIMD architecture would seem to be ideal given the structure of a process system as a network of interconnected components. Our candidate computer for further research and development of both AKG and real-time frame-based diagnosers is the Connection Machine™ (CM) built by Thinking Machines, Inc. of Cambridge, Massachusetts. The team has recently received a *lisp simulator for the machine (*lisp is a parallel lisp that uses the parallel processing features of the CM) and we will be examining the feasibility of implementing AKG in parallel with it.

Our intermediate goals (for the second year, first quarter) are:

a) Finish CAD database for OMRF, develop component and to-from lists consistent with AKG requirements.

b) Add CAD graphics using ANSI standards to the Graphic User Interface.

c) Evaluate Connection Machine implementation of AKG using the simulator, and if warranted, prepare a proposal for acquisition of hardware.

6.0 Acknowledgements:

The Authors would like to recognize Messers. Massood Towhidnejad and Frederic McKenzie for assisting in the preparation of this document, and welcome Ms. Robin Kladke as the newest member of the research team.

```
(deframe COMPRESSOR-1
                (nomenclature "Pnuematic Compressor #1")
                (aio compressor)
                (source a-t-e-6-25M)
                (status (cond ((< a-t-e-6-25m .5)
                               0.0)
                              ((< a-t-e-6-25m 1.9)
                               (+ (* a-t-e-6-25m 3.93) -1.96))
                              ((< a-t-e-6-25m 4)
                               (+ (* a-t-e-6-25m 8.81) -11.24))
                              ((< a-t-e-6-25m 6.001)
                               (+ (* a-t-e-6-25m 6.66) -2.67))))
                (in-path-of output-pressure1)
                (delay 660)
                (units "psig"))


(deframe A-T-E-36-8M
                (nomenclature "Power supply #2")
                (aio analog-power-supply)
                (source-path da-out-b)
                (status  (* da-out-b 3.6))
                (sinks compressor-2)
                (in-path-of analog-in-1)
                (units "volts"))

(deframe K1
                (aio relay)
                (nomenclature "Shut-off Valve Controll Relay")
                (source-path device1)
                (in-path-of v27200-54))


(deframe DEVICE3
                (aio not-gate)
                (nomenclature "Shut-off Valve Primary Select Gate")
                (source-path (not ttl-out-1))
                (in-path-of device2))

(deframe TTL-OUT-1
          (aio ttl-output)
          (aeo TTL-OUT-CARD1)
          (address 1)
          (in-path-of)
          (cvalue off))

(f-replace 'ttl-out-0 'in-path-of 'device2)
(f-replace 'ttl-out-1 'in-path-of 'device3)
(fput 'ttl-out-1 'in-path-of 'device4)
(f-replace 'ttl-out-2 'in-path-of 'device4)
```

Figure 1. Human generated frames.

```
(deframe COMPRESSOR#1
         (nomenclature "Pnuematic Compressor #1")
         (aio COMPRESSOR)
         (source-path (ATE6-25M))
         (status (COND ((< (CSTATUS ATE6-25M) 0.5) 0.0)
        ((< (CSTATUS ATE6-25M) 1.9) (+ (* (CSTATUS ATE6-25M) 3.93) -1.96))
        ((< (CSTATUS ATE6-25M) 4) (+ (* (CSTATUS ATE6-25M) 8.81) -11.24))
        ((< (CSTATUS ATE6-25M) 6.001) (+ (* (CSTATUS ATE6-25M) 6.66) -2.67)))
         (in-path-of (MV-74))
         (delay 660)
         (range 0-40)
         (units "PSI"))


(deframe ATE36-8M
         (nomenclature "Power supply #2")
         (aio ANALOG-POWER-SUPPLY)
         (source-path (DA-OUT-B))
         (status (* (STATVAL 'DA-OUT-B) 3.6))
         (in-path-of (ANALOG-IN-1 COMPRESSOR#2))
         (range 0-36)
         (units "VDC"))

(deframe K1
         (nomenclature "Shut-off valve controll relay")
         (aio RELAY)
         (source-path (DEVICE-1 PWR-2))
         (in-path-of (V27200-54 PWR-1R)))


(deframe DEVICE-3
         (nomenclature "Shut-off valve primary select gate")
         (aio NOT-GATE)
         (source-path (PWR-1R TTL-OUT-1))
         (in-path-of (PWR-1 DEVICE-2)))

(deframe TTL-OUT-1
         (aio TTL-OUTPUT)
         (aeo TTL-OUT-CARD1)
         (source T)
         (cvalue OFF)
         (in-path-of (DEVICE-3 DEVICE-4)))
```

Figure 2. AKG generated frames.

```
un TTL-INPUT (&rest frames &aux changed-list)
et* ((addr-string (car frames))
     (bit-list (bi-read addr-string))        ;initial read
     (time (timer)))
(dolist (measurement-frame (cdr frames))
  (let ((current-value (unboolify (nth (car (mget1 measurement-frame
                                                    'address))
                                       bit-list)))
        (frame-value (car (mget1 measurement-frame 'cvalue))))
    (when (not (equal current-value frame-value))
      (push (list measurement-frame current-value time) changed-list))))
hanged-list))

n ANALOG-INPUT (&rest frames &aux changed-list)
t* ((addr-string (car frames)))
dolist (measurement-frame (cdr frames))
  (let* ((current-value (analog-read                    ;read the latest value
                         addr-string
                         (car (mget1 measurement-frame 'address))))
         (time (timer))
         (frame-value (car (mget1 measurement-frame 'cvalue))))
    (unless (<= (- frame-value 0.03) current-value (+ frame-value 0.03))
      ;;hard-coded accuract of 0.06 volts dc        !!!
      (push (list measurement-frame
                  (rounder current-value 2) time) changed-list)))))
hanged-list)

n ANALOG-READ (addr-string channel &aux (string ""))
wrt (format nil "~aRS"                    ; RESET SEQUENCE STACK command
            addr-string))
wrt (format nil "~aS1,~a"                 ; SEQUENCE command, uses format #1
            addr-string channel))         ; (see k-init.lsp/setup-measurements)
wrt (format nil "~aTF1"                   ; TRIGGER command,
            addr-string))                 ; 1 meas cycle with max sample time
wrt (format nil "~aAA1"                   ; ACCEPT command, gets the card ready
            addr-string))                 ; for 1 measurement in ascii format
:%sysint #x60 1 0 6 0)                    ;IBRD function (6 bytes=1 meas'mt)
imes (byte 6)
ultiple-value-bind (byte word n-word)
 (sys:%contents *1-data-seg-p* byte)
 (setf string (string-append string (format nil "~c" byte)))))
d-from-string string))                    ;so it'll return a number
```

Figure 3. Functions in the human generated Knowledge base